Auffinden von Spaltenkorrelationen mithilfe proaktiver und reaktiver Verfahren

Katharina Büchse
Friedrich-Schiller-Universität
Institut für Informatik
Ernst-Abbe-Platz 2
07743 Jena
katharina.buechse@uni-jena.de

KURZFASSUNG

Zur Verbesserung von Statistikdaten in relativen Datenbanksystemen werden seit einigen Jahren Verfahren für das Finden von Korrelationen zwischen zwei oder mehr Spalten entwickelt. Dieses Wissen über Korrelationen ist notwendig, weil der Optimizer des Datenbankmanagementsystems (DBMS) bei der Anfrageplanerstellung sonst von Unabhängigkeit der Daten ausgeht, was wiederum zu groben Fehlern bei der Kostenschätzung und somit zu schlechten Ausführungsplänen führen kann.

Die entsprechenden Verfahren gliedern sich grob in proaktive und reaktive Verfahren: Erstere liefern ein gutes Gesamtbild über sämtliche vorhandenen Daten, müssen dazu allerdings selbst regelmäßig auf die Daten zugreifen und benötigen somit Kapazität des DBMS. Letztere überwachen und analysieren hingegen die Anfrageergebnisse und liefern daher nur Korrelationsannahmen für bereits abgefragte Daten, was einerseits das bisherige Nutzerinteresse sehr gut widerspiegelt, andererseits aber bei Änderungen des Workloads versagen kann. Dafür wird einzig bei der Überwachung der Anfragen DBMS-Kapazität benötigt, es erfolgt kein eigenständiger Zugriff auf die Daten.

Im Zuge dieser Arbeit werden beide Ansätze miteinander verbunden, um ihre jeweiligen Vorteile auszunutzen. Dazu werden die sich ergebenden Herausforderungen, wie sich widersprechende Korrelationsannahmen, aufgezeigt und als Lösungsansatz u. a. der zusätzliche Einsatz von reaktiv erstellten Statistiken vorgeschlagen.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management; H.2.4 [**Database Management**]: Systems—*Query processing*

General Terms

Theory, Performance

25^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 28.05.2013 - 31.05.2013, Ilmenau, Germany.
Copyright is held by the author/owner(s).

Keywords

Anfrageoptimierung, Spaltenkorrelation, Feedback

1. EINFÜHRUNG

Die Verwaltung großer Datenmengen benötigt zunehmend leistungsfähigere Algorithmen, da die Verbesserung der Technik (Hardware) nicht mit dem immer höheren Datenaufkommen heutiger Zeit mithalten kann. Bspw. werden wissenschaftliche Messergebnisse aufgrund besserer Messtechnik immer genauer und umfangreicher, sodass Wissenschaftler sie detaillierter, aber auch umfassender analysieren wollen und müssen, oder Online-Shops speichern sämtliche ihrer Verkaufsdaten und werten sie aus, um dem Benutzer passend zu seinen Interessen zeitnah und individuell neue Angebote machen zu können.

Zur Verwaltung dieser wie auch anderer Daten sind (im Datenbankbereich) insbesondere schlaue Optimizer gefragt, weil sie für die Erstellung der Anfragepläne (und somit für die Ausführungszeit einer jeden Anfrage) verantwortlich sind. Damit sie in ihrer Wahl nicht völlig daneben greifen, gibt es Statistiken, anhand derer sie eine ungefähre Vorstellung bekommen, wie die vorhandene Datenlandschaft aussieht. Hierbei ist insbesondere die zu erwartende Tupelanzahl von Interesse, da sie in hohem Maße die Ausführungszeit einer Anfrage beeinflusst. Je besser die Statistiken die Verteilung der Daten wiedergeben (und je aktueller sie sind), desto besser ist der resultierende Ausführungsplan. Sind die Daten unkorreliert (was leider sehr unwahrscheinlich ist), genügt es, pro zu betrachtender Spalte die Verteilung der Werte innerhalb dieser Spalte zu speichern. Treten in diesem Fall später in den Anfragen Kombinationen der Spalten auf, ergibt sich die zu erwartende Tupelanzahl mithilfe einfacher statistischer Weisheiten (durch Multiplikation der Einzelwahrscheinlichkeiten).

Leider versagen diese ab einem bestimmten Korrelationsgrad (also bei korrelierten Daten), und zwar in dem Sinne, dass die vom Optimizer berechneten Schätzwerte zu stark von der Wirklichkeit abweichen, was wiederum zu schlechten Ausführungszeiten führt. Diese ließen sich u.U. durch die Wahl eines anderen Plans, welcher unter Berücksichtigung der Korrelation vom Optimizer erstellt wurde, verringern oder sogar vermeiden.

Zur Veranschaulichung betrachten wir eine Tabelle, welche u. a. die Spalten A und B besitzt, und eine Anfrage, welche Teile eben dieser Spalten ausgeben soll. Desweiteren liege auf Spalte B ein Index, den wir mit I_B bezeichnen wol-

len, und es existiere ein zusammengesetzter Index $I_{A,B}$ für beide Spalten. Beide Indizes seien im DBMS mithilfe von Bäumen (bspw. B*-Bäume) implementiert, sodass wir auch (etwas informell) von „flachen“ oder „hohen“ Indizes sprechen können.

Sind beide Spalten unkorreliert, so lohnt sich in der Regel die Abfrage über $I_{A,B}$. Bei einer starken Korrelation beider Spalten dagegen könnte die alleinige Verwendung von I_B vorteilhaft sein, und zwar wenn die Werte aus Spalte A i.d.R. durch die Werte aus Spalte B bestimmt werden (ein typisches Beispiel, welches auch in CORDS [7] anzutreffen ist, wäre eine Tabelle „Auto“ mit den Spalten A = „Firma“ und B = „Marke“, sodass sich für A Werte wie „Opel“ oder „Mercedes“ und für B Werte wie „Zafira“ oder „S-Klasse“ ergeben). Statt nun im vergleichsweise hohen Index $I_{A,B}$ erst passende A- und dann passende B-Werte zu suchen, werden sämtliche Tupel, welche die gewünschten B-Werte enthalten, über den flacheren Index I_B geladen und überprüft, ob die jeweiligen A-Werte der Anfrage entsprechen (was aufgrund der Abhängigkeit der Regelfall sein sollte).

Das Wissen über Korrelationen fällt aber natürlich nicht vom Himmel, es hat seinen Preis. Jeder Datenbänkler hofft, dass seine Daten unkorreliert sind, weil sein DBMS dann weniger Metadaten (also Daten über die Daten) speichern und verwalten muss, sondern auf die bereits erwähnten statistischen Weisheiten zurückgreifen kann. Sind die Daten dagegen (stark) korreliert, lässt sich die Erkenntnis darüber nicht so einfach wie die Unabhängigkeit mit (anderen) statistischen Weisheiten verbinden und somit „abarbeiten“.

Nicht jede (eher kaum eine) Korrelation stellt eine (schwache) funktionale Abhängigkeit dar, wie es im Beispiel der Fall war, wo wir einfach sagen konnten „Aus der Marke folgt die Firma (bis auf wenige Ausnahmen)“. Oft liebäugeln bestimmte Werte der einen Spalte mit bestimmten Werten anderer Spalten, ohne sich jedoch in irgendeiner Weise auf diese Kombinationen zu beschränken. (In Stuttgart gibt es sicherlich eine Menge Porsches, aber die gibt es woanders auch.) Außerdem ändern sie möglicherweise mit der Zeit ihre Vorlieben (das Stuttgarter Porschewerk könnte bspw. nach China umziehen) oder schaffen letztere völlig ab (wer braucht schon einen Porsche? Oder überhaupt ein Auto?).

Deswegen werden für korrelierte Daten zusätzliche Statistiken benötigt, welche nicht nur die Werteverteilung einer, sondern die Werteverteilung mehrerer Spalten wiedergeben. Diese zusätzlichen Statistiken müssen natürlich irgendwo abgespeichert und, was noch viel schlimmer ist, gewartet werden. Somit ergeben sich zusätzlicher Speicherbedarf und zusätzlicher Aufwand, also viel zu viel von dem, was keiner so richtig will.

Da sich ein bisschen statistische Korrelation im Grunde überall findet, gilt es, die Korrelationen ausfindig zu machen, welche unsere statistischen Weisheiten alt aussehen lassen und dazu führen, dass das Anfrageergebnis erst nach einer gefühlten halben Ewigkeit ausgeben wird. Ob letzteres überhaupt passiert, hängt natürlich auch vom Anfrageverhalten auf die Datenbank ab. Wenn die Benutzer sich in ihren (meist mithilfe von Anwendungsprogrammen abgesetzten) SQL-Anfragen in der WHERE-Klausel jeweils auf eine Spalte beschränken und auf jedwede Verbünde (Joins) verzichten, dann ist die Welt in Ordnung. Leider lassen sich Benutzer nur ungern so stark einschränken.

Daher gibt es zwei grundsätzliche Möglichkeiten: Entweder schauen wir dem Benutzer auf die Finger und suchen in den von ihm abgefragten Daten nach Korrelationen (das entspricht einer reaktiven Vorgehensweise), oder wir „suchen uns selbst“ ein paar Daten der Datenbank „aus“, die wir untersuchen wollen (und gehen somit proaktiv vor). Beide Vorgehensweisen haben ihre Vor- und Nachteile. Während im reaktiven Fall keine Daten speziell zur Korrelationsfindung „angefasst“ werden müssen, hier aber alle Daten, die bis zu einer bestimmten Anfrage nie abgefragt wurden, als unkorreliert gelten, müssen wir für die proaktive Methode (also nur zum Feststellen, ob Korrelation vorherrscht) extra Daten lesen, sind aber für (fast) alle Eventualitäten gewappnet.

Interessanterweise kann es vorkommen, dass beide Methoden für ein und dieselbe Spaltenkombination unterschiedliche Ergebnisse liefern (der Einfachheit halber beschränken wir uns hierbei auf die möglichen Ergebnisse „korreliert“ oder „unabhängig“). Für den Fall, dass die reaktive Methode eine Spaltenkombination gar nicht betrachtet hat, sollte das klar sein. Aber nehmen wir an, dass die Kombination von beiden Methoden analysiert wurde. Da für die Analyse höchstwahrscheinlich jeweils unterschiedliche Tupel (Wertekombinationen) verwendet wurden, können sich natürlich auch die Schlüsse unterscheiden. Hier stellt sich nun die Frage, welches Ergebnis „besser“ ist. Dafür gibt es keine allgemeine Antwort, gehen wir aber von einer moderaten Änderung des Anfrageverhaltens aus, ist sicherlich das „reaktive Ergebnis“ kurzfristig entscheidender, während das „proaktive Ergebnis“ in die längerfristige Planung der Statistikerstellung mit aufgenommen werden sollte.

2. GRUNDLAGEN

Wie in der Einleitung bereits angedeutet, können Korrelationen einem Datenbanknutzer den Tag vermiesen. Um dies zu verhindern, wurden einige Methoden vorgeschlagen, welche sich auf verschiedene Art und Weise dieser Problematik annehmen (z. B. [7, 6]) oder sie sogar ausnutzen (z. B. [4, 8]), um noch an Performance zuzulegen. Letztere sind allerdings mit hohem Aufwand oder der Möglichkeit, fehlerhafte Anfrageergebnisse zu liefern[1], verbunden. Daher konzentrieren wir uns hier auf das Erkennen von Korrelationen allein zur Verbesserung der Statistiken und wollen hierbei zwischen proaktiven und reaktiven Verfahren unterscheiden.

2.1 Proaktive (datengetriebene) Verfahren

Proaktiv zu handeln bedeutet, etwas „auf Verdacht“ zu tun. Impfungen sind dafür ein gutes Beispiel – mithilfe einer Impfung ist der Körper in der Lage, Krankheitserreger zu bekämpfen, aber in vielen Fällen ist unklar, ob er diese Fähigkeit jemals benötigen wird. Da Impfungen auch mit Nebenwirkungen verbunden sein können, muss jeder für sich entscheiden, ob und wogegen er sich impfen lässt.

Auch Datenbanken können „geimpft“ werden, allerdings handelt es sich bei langen Anfrageausführungszeiten (die wir ja bekämpfen wollen) eher um Symptome (wie Bauchschmerzen oder eine laufende Nase), die natürlich unterschiedliche Ursachen haben können. Eine davon bilden ganz

[1]Da die Verfahren direkt in die Anfrageplanerstellung eingreifen und dabei auf ihr Wissen über Korrelationen aufbauen, muss, für ein korrektes Anfrageergebnis, dieses Wissen aktuell und vollständig sein.

klar Korrelationen zwischen den Daten, wobei natürlich erst ein gewisses Maß an Korrelation überhaupt als „krankhaft“ anzusehen ist. (Es benötigt ja auch eine gewisse Menge an Bakterien, damit eine Krankheit mit ihren Symptomen ausbricht.) Der grobe „Impfvorgang“ „gegen“ Korrelationen umfasst zwei Schritte:

1. Es werden Vermutungen aufgestellt, welche Spaltenkombinationen für spätere Anfragen eine Rolle spielen könnten.

2. Es wird kontrolliert, ob diese Kombinationen von Korrelation betroffen sind oder nicht.

Entscheidend dabei ist, dass die Daten bzw. ein Teil der Daten gelesen (und analysiert) werden, und zwar ohne damit konkrete Anfragen zu bedienen, sondern rein zur Ausführung des Verfahrens bzw. der „Impfung“ (in diesem Fall „gegen“ Korrelation, wobei die Korrelation natürlich nicht beseitigt wird, schließlich können wir schlecht den Datenbestand ändern, sondern die Datenbank lernt, damit umzugehen). Das Lesen und Analysieren kostet natürlich Zeit, womit klar wird, dass auch diese „Impfung“ „Nebenwirkungen“ mit sich bringt.

Eine konkrete Umsetzung haben Ilyas et al., aufbauend auf BHUNT [4], mit CORDS [7] vorgestellt. Dieses Verfahren findet Korrelationen zwischen Spaltenpaaren, die Spaltenanzahl pro Spaltenkombination wurde also auf zwei begrenzt.[2]

Es geht folgendermaßen vor: Im ersten „Impfschritt“ sucht es mithilfe des Katalogs oder mittels Stichproben nach Schlüssel-Fremdschlüssel-Beziehungen und führt somit eine Art Rückabbildung von Datenbank zu Datenmodell durch (engl. „reverse engineering“) [4]. Darauf aufbauend werden dann nur solche Spaltenkombinationen als für die Korrelationssuche infrage kommend angesehen, deren Spalten

a) aus derselben Tabelle stammen oder

b) aus einer Verbundtabelle stammen, wobei der Verbund („Join“) mittels (Un-) Gleichung zwischen Schlüssel- und Fremdschlüsselspalten entstanden ist.

Zudem gibt es zusätzliche Reduktionsregeln (engl. „pruning rules“) für das Finden der Beziehungen und für die Auswahl der zu betrachtenden Spaltenkombinationen. Schließlich kann die Spaltenanzahl sehr hoch sein, was die Anzahl an möglichen Kombinationen gegebenenfalls ins Unermessliche steigert.

Im zweiten „Impfschritt“ wird für jede Spaltenkombination eine Stichprobe entnommen und darauf aufbauend eine Kontingenztabelle erstellt. Letztere dient dann wiederum als Grundlage für einen Chi-Quadrat-Test, der als Ergebnis eine Zahl $\chi^2 \geq 0$ liefert. Gilt $\chi^2 = 0$, so sind die Spalten vollständig unabhängig. Da dieser Fall aber in der Praxis kaum auftritt, muss χ^2 einen gewissen Schwellwert überschreiten, damit die entsprechende Spaltenkombination als korreliert angesehen wird. Zum Schluss wird eine Art Rangliste der Spaltenkombinationen mit den höchsten χ^2-Werten erstellt und für die obersten n Kombinationen werden zusätzliche Statistikdaten angelegt. Die Zahl n ist dabei u. a. durch die Größe des Speicherplatzes (für Statistikdaten) begrenzt.

[2]Die Begrenzung wird damit begründet, dass auf diese Weise das beste Aufwand-Nutzen-Verhältnis entsteht. Das Verfahren selbst ist nicht auf Spaltenpaare beschränkt.

2.2 Reaktive (anfragegetriebene) Verfahren

Während wir im vorherigen Abschnitt Vermutungen aufgestellt und auf Verdacht gehandelt haben, um den Datenbankbenutzer glücklich zu machen, gehen wir jetzt davon aus, dass den Benutzer auch weiterhin das interessieren wird, wofür er sich bis jetzt interessiert hat.

Wir ziehen also aus der Vergangenheit Rückschlüsse für die Zukunft, und zwar indem wir den Benutzer bei seinem Tun beobachten und darauf reagieren (daher auch die Bezeichnung „reaktiv“). Dabei achten wir nicht allein auf die gestellten SQL-Anfragen, sondern überwachen viel mehr die von der Datenbank zurückgegebenen Anfrageergebnisse. Diese verraten uns nämlich alles (jeweils 100-prozentig aktuell!) über den Teil der vorhandenen Datenlandschaft, den der Benutzer bis jetzt interessant fand.

Auf diese Weise können bspw. Statistiken erzeugt werden [5, 11, 3] (wobei STHoles [5] und ISOMER [11] sogar in der Lage sind, mehrdimensionale Statistiken zu erstellen) oder es lassen sich mithilfe alter Anfragen neue, ähnliche Anfragen in ihrer Performance verbessern [12]. Sinnvoll kann auch eine Unterbrechung der Anfrageausführung mit damit verbundener Reoptimierung sein [9, 2, 10]. Zu guter letzt lässt sich mithilfe dieses Ansatzes zumindest herausfinden, welche Statistikdaten entscheidend sein könnten [1].

In [1] haben Aboulnaga et al. auch schon erste Ansätze für eine Analyse auf Spaltenkorrelation vorgestellt, welche später in [6] durch Haas et al. ausgebaut und verbessert wurden. In Analogie zu CORDS werden in [1] und [6] nur Spaltenpaare für die Korrelationssuche in Betracht gezogen. Allerdings fällt die Auswahl der infrage kommenden Spaltenpaare wesentlich leichter aus, weil einfach alle Spaltenpaare, die in den Anfragen (mit hinreichend vielen Daten[3]) vorkommen, potentielle Kandidaten bilden.

Während in [1] pro auftretendes Wertepaar einer Spaltenkombination ein Quotient aus „Häufigkeit bei Unabhängigkeit“ und „tatsächliche Häufigkeit“ gebildet und das Spaltenpaar als „korreliert“ angesehen wird, sobald zu viele dieser Quotienten von einem gewissen Wert abweichen, setzen Haas et al. in [6] einen angepassten Chi-Quadrat-Test ein, um Korrelationen zu finden. Dieser ist etwas aufwendiger als die Vorgehensweise von [1], dafür jedoch nicht so fehleranfällig [6]. Zudem stellen Haas et al. in [6] Möglichkeiten vor, wie sich die einzelnen „Korrelationswerte“ pro Spaltenpaar miteinander vergleichen lassen, sodass, ähnlich wie in CORDS, eine Rangliste der am stärksten korrelierten Spaltenpaare erstellt werden kann. Diese kann als Entscheidungshilfe für das Anlegen zusätzlicher Statistikdaten genutzt werden.

3. HERAUSFORDERUNGEN

In [6] wurde bereits vorgeschlagen, dieses Verfahren mit CORDS zu verbinden. Das reaktive Verfahren spricht aufgrund seiner Effizienz für sich, während das proaktive Verfahren eine gewisse Robustheit bietet und somit bei Lernphasen von [6] (wenn es neu eingeführt wird oder wenn sich die Anfragen ändern) robuste Schätzwerte zur Erstellung eines Anfrageplans berechnet werden können [6]. Dazu sollte CORDS entweder in einem gedrosselten Modus während des normalen Datenbankbetriebs laufen oder während Wartungszeiten ausgeführt werden. Allerdings werden in [6] keine Aussagen darüber getroffen, wie die jeweiligen Ergebnis-

[3]Um aussagefähige Ergebnisse zu bekommen, wird ein gewisses Mindestmaß an Beobachtungen benötigt, insb. in [6].

se beider Verfahren miteinander kombiniert werden sollten. Folgende Punkte sind dabei zu bedenken:

- Beide Verfahren liefern eine Rangliste mit den als am stärksten von Korrelation betroffenen Spalten. Allerdings sind die den Listen zugrunde liegenden „Korrelationswerte“ (s. bspw. χ^2 im Abschnitt über proaktive Verfahren) auf unterschiedliche Weise entstanden und lassen sich nicht einfach vergleichen. Liefern beide Listen unterschiedliche Spaltenkombinationen, so kann es passieren, dass eine Kombination, die in der eine Liste sehr weit unten erscheint, stärker korreliert ist, als Kombinationen, die auf der anderen Liste sehr weit oben aufgeführt sind.
- Die Daten, welche zu einer gewissen Entscheidung bei den beiden Verfahren führen, ändern sich, werden aber in der Regel nicht gleichzeitig von beiden Verfahren gelesen. Das hängt damit zusammen, dass CORDS zu einem bestimmten Zeitpunkt eine Stichprobe entnimmt und darauf seine Analyse aufbaut, während das Verfahren aus [6] die im Laufe der Zeit angesammelten Anfragedaten auswertet.
- Da zusätzliche Statistikdaten Speicherplatz benötigen und vor allem gewartet werden müssen, ist es nicht sinnvoll, einfach für alle Spaltenkombinationen, die in der einen und/oder der anderen Rangliste vorkommen, gleich zu verfahren und zusätzliche Statistiken zu erstellen.

Zur Verdeutlichung wollen wir die Tabelle aller Firmenwagen eines großen, internationalen IT-Unternehmens betrachten, in welcher zu jedem Wagen u. a. seine Farbe und die Personal- sowie die Abteilungsnummer desjenigen Mitarbeiters verzeichnet ist, der den Wagen hauptsächlich nutzt. Diverse dieser Mitarbeiter wiederum gehen in einem Dresdener mittelständischen Unternehmen ein und aus, welches nur rote KFZ auf seinem Parkplatz zulässt (aus Kapazitätsgründen wurde eine solche, vielleicht etwas seltsam anmutende Regelung eingeführt). Da die Mitarbeiter sich dieser Regelung bei der Wahl ihres Wagens bewusst waren, fahren sie alle ein rotes Auto. Zudem sitzen sie alle in derselben Abteilung.

Allerdings ist das internationale Unternehmen wirklich sehr groß und besitzt viele Firmenwagen sowie unzählige Abteilungen, sodass diese roten Autos in der Gesamtheit der Tabelle nicht auffallen. In diesem Sinne würde das proaktive Verfahren CORDS also (eher) keinen Zusammenhang zwischen der Abteilungsnummer des den Wagen benutzenden Mitarbeiters und der Farbe des Autos erkennen.

Werden aber häufig genau diese Mitarbeiter mit der Farbe ihres Wagens abgefragt, z. B. weil sich diese kuriose Regelung des mittelständischen Unternehmens herumspricht, es keiner so recht glauben will und deswegen die Datenbank konsultiert, so könnte ein reaktives Verfahren feststellen, dass beide Spalten korreliert sind. Diese Feststellung tritt insbesondere dann auf, wenn sonst wenig Anfragen an beide betroffenen Spalten gestellt werden, was durchaus möglich ist, weil sonst die Farbe des Wagens eine eher untergeordnete Rolle spielt.

Insbesondere der letztgenannte Umstand macht deutlich, dass es nicht sinnvoll ist, Statistikdaten für die Gesamtheit beider Spalten zu erstellen und zu warten. Aber die Tatsache, dass bestimmte Spezialfälle für den Benutzer besonders interessant sein könnten, die möglicherweise eben gerade mit Korrelation einhergehen, spricht wiederum für eine Art „Hinweis“ an den Optimizer.

4. LÖSUNGSANSATZ

Da CORDS wie auch das Verfahren aus [6] nur Spaltenpaare betrachten und dies mit einem sich experimentell ergebenem Aufwand-Nutzen-Optimum begründen, werden auch wir uns auf Spaltenpaare begrenzen. Allerdings wollen wir uns für die Kombination von proaktiver und reaktiver Korrelationssuche zunächst nicht auf diese beiden Verfahren beschränken, müssen aber doch gewisse Voraussetzungen an die verwendeten Verfahren (und das Datenmodell der Datenbank) stellen. Diese seien hier aufgezählt:

1. Entscheidung über die zu untersuchenden Spaltenkombinationen:
   - Das proaktive Verfahren betreibt „reverse engineering“, um zu entscheiden, welche Spaltenkombinationen untersucht werden sollen.
   - Das Datenmodell der Datenbank ändert sich nicht, bzw. sind nur geringfügige Änderungen zu erwarten, welche vom proaktiven Verfahren in das von ihm erstellte Datenmodell sukzessive eingearbeitet werden können. Auf diese Weise können wir bei unseren Betrachtungen den ersten „Impfschritt“ vernachlässigen.
2. Datengrundlage für die Untersuchung:
   - Das proaktive Verfahren entnimmt für jegliche zu untersuchende Spaltenkombination eine Stichprobe, welche mit einem Zeitstempel versehen wird. Diese Stichprobe wird solange aufbewahrt, bis das Verfahren auf „Unkorreliertheit“ plädiert oder für die entsprechende Spaltenkombination eine neue Stichprobe erstellt wird.
   - Das reaktive Verfahren bedient sich eines Query-Feedback-Warehouses, in welchem die Beobachtungen („Query-Feedback-Records“) der Anfragen notiert sind.
3. Vergleich der Ergebnisse:
   - Beide Verfahren geben für jede Spaltenkombination, die sie untersucht haben, einen „Korrelationswert“ aus, der sich innerhalb des Verfahrens vergleichen lässt. Wie dieser genau berechnet wird, ist für uns unerheblich.
   - Aus den höchsten Korrelationswerten ergeben sich zwei Ranglisten der am stärksten korrelierten Spaltenpaare, die wir unterschiedlich auswerten wollen.

Zudem wollen wir davon ausgehen, dass das proaktive Verfahren in einem gedrosselten Modus ausgeführt wird und somit sukzessive seine Rangliste befüllt. (Zusätzliche Wartungszeiträume, bei denen das Verfahren ungedrosselt laufen kann, beschleunigen die Arbeit und bilden somit einen schönen Zusatz, aber da heutzutage viele Datenbanken quasi dauerhaft laufen müssen, wollen wir sie nicht voraussetzen.)

Das reaktive Verfahren dagegen wird zu bestimmten Zeitpunkten gestartet, um die sich bis dahin angesammelten Beobachtungen zu analysieren, und gibt nach beendeter Analyse seine Rangliste bekannt. Da es als Grundlage nur die Daten aus dem Query-Feedback-Warehouse benötigt, kann es völlig entkoppelt von der eigentlichen Datenbank laufen.

Ist die reaktive Rangliste bekannt, kann diese mit der (bis dahin angefertigten) proaktiven Rangliste verglichen werden. Tritt eine Spaltenkombination in beiden Ranglisten auf, so bedeutet das, dass diese Korrelation für die bisherigen Anfragen eine Rolle gespielt hat und nicht nur auf Einzelfälle beschränkt ist, sondern auch mittels Analyse einer repräsentativen Stichprobe an Wertepaaren gefunden wurde.

Unter diesen Umständen lassen wir mittels einer Stichprobe Statistikdaten für die betreffende Spaltenkorrelation erstellen. Dabei wählen wir die Stichprobe des proaktiven Verfahrens, solange diese ein gewisses Alter nicht überschritten hat. Ist sie zu alt, wird eine neue Stichprobe entnommen.[4]

Interessanter wird es, wenn nur eines der Verfahren auf Korrelation tippt, während das andere Verfahren die entsprechende Spaltenkombination nicht in seiner Rangliste enthält. Die Ursache dafür liegt entweder darin, dass letzteres Verfahren die Kombination noch nicht analysiert hat (beim reaktiven Verfahren heißt das, dass sie nicht oder zu selten in den Anfragen vorkam), oder bei seiner Analyse zu dem Ergebnis „nicht korreliert“ gekommen ist.

Diese Unterscheidung wollen wir insbesondere in dem Fall vornehmen, wenn einzig das reaktive Verfahren die Korrelation „entdeckt“ hat. Unter der Annahme, dass weitere, ähnliche Anfragen folgen werden, benötigt der Optimizer schnell Statistiken für den abgefragten Bereich. Diese sollen zunächst reaktiv mithilfe der Query-Feedback-Records aus der Query-Feedback-Warehouse erstellt werden (unter Verwendung von bspw. [11], wobei wir nur zweidimensionale Statistiken benötigen). Das kann wieder völlig getrennt von der eigentlichen Datenbank geschehen, da nur das Query-Feedback-Warehouse als Grundlage dient.

Wir überprüfen nun, ob das proaktive Verfahren das Spaltenpaar schon bearbeitet hat. Dies sollte anhand der Abarbeitungsreihenfolge der infrage kommenden Spaltenpaare erkennbar sein.

Ist dem so, hat das proaktive Verfahren das entsprechende Paar als „unkorreliert“ eingestuft und wir bleiben bei den reaktiv erstellten Statistiken, die auch nur reaktiv aktualisiert werden. Veralten sie später zu stark aufgrund fehlender Anfragen (und somit fehlendem Nutzerinteresse), können sie gelöscht werden.

Ist dem nicht so, geben wir die entsprechende Kombination an das proaktive Verfahren weiter mit dem Auftrag, diese zu untersuchen.[5] Beim nächsten Vergleich der Ranglisten muss es für das betrachtete Spaltenpaar eine konkrete Antwort geben. Entscheidet sich das proaktive Verfahren für „korreliert“ und befindet sich das Spaltenpaar auch wieder in der Rangliste des reaktiven Verfahrens, dann löschen wir die reaktiv erstellten Statistiken und erstellen neue Statistiken mittels einer Stichprobe, analog zum ersten Fall. (Die Kombination beider Statistiktypen wäre viel zu aufwendig, u. a. wegen unterschiedlicher Entstehungszeitpunkte.) Wenn das proaktive Verfahren dagegen explizit „unkorreliert“ ausgibt, bleibt es bei den reaktiv erstellten Statistiken, s. oben.

Wenn jedoch nur das proaktive Verfahren eine bestimmte Korrelation erkennt, dann ist diese Erkenntnis zunächst für die Benutzer unerheblich. Sei es, weil der Nutzer diese Spaltenkombination noch gar nicht abgefragt hat, oder weil er bis jetzt nur den Teil der Daten benötigt hat, der scheinbar unkorreliert ist. In diesem Fall markieren wir nur im Datenbankkatalog (wo die Statistiken abgespeichert werden) die beiden Spalten als korreliert und geben dem Optimizer somit ein Zeichen, dass hier hohe Schätzfehler möglich sind und er deswegen robuste Pläne zu wählen hat. Dabei bedeutet „robust“, dass der gewählte Plan für die errechneten Schätzwerte möglicherweise nicht ganz optimal ist, dafür aber bei stärker abweichenden „wahren Werten“ immer noch akzeptable Ergebnisse liefert. Zudem können wir ohne wirklichen Einsatz des reaktiven Verfahrens die Anzahl der Anfragen zählen, die auf diese Spalten zugreifen und bei denen sich der Optimizer stark verschätzt hat. Übersteigt der Zähler einen Schwellwert, werden mithilfe einer neuen Stichprobe (vollständige, also insb. mit Werteverteilung) Statistikdaten erstellt und im Katalog abgelegt.

Der Vollständigkeit halber wollen wir hier noch den Fall erwähnen, dass eine Spaltenkombination weder in der einen, noch in der anderen Rangliste vorkommt. Es sollte klar sein, dass diese Kombination als „unkorreliert“ angesehen und somit für die Statistikerstellung nicht weiter betrachtet wird.

5. AUSBLICK

Die hier vorgestellte Vorgehensweise zur Verbesserung der Korrelationsfindung mittels Einsatz zweier unterschiedlicher Verfahren muss weiter vertieft und insbesondere praktisch umgesetzt und getestet werden. Vor allem muss ein passendes Datenmodell für die reaktive Erstellung von Spaltenpaarstatistiken gefunden werden. Das vorgeschlagene Verfahren ISOMER [11] setzt hier auf STHoles [5], einem Datenmodell, welches bei sich stark überschneidenden Anfragen schnell inperformant werden kann. Für den eindimensionalen Fall wurde bereits von Informix-Entwicklern eine performante Lösung vorgestellt [3], welche sich aber nicht einfach auf den zweidimensionalen Fall übertragen lässt.

Eine weitere, noch nicht völlig ausgearbeitete Herausforderung bildet die Tatsache, dass das proaktive Verfahren im gedrosselten Modus läuft und erst sukzessive seine Rangliste erstellt. Das bedeutet, dass wir eigentlich nur Zwischenergebnisse dieser Rangliste mit der reaktiv erstellten Rangliste vergleichen. Dies kann zu unerwünschten Effekten führen, z. B. könnten beide Ranglisten völlig unterschiedliche Spaltenkombinationen enthalten, was einfach der Tatsache geschuldet ist, dass beide Verfahren unterschiedliche Spaltenkombinationen untersucht haben. Um solche Missstände zu vermeiden, muss die proaktive Abarbeitungsreihenfolge der Spaltenpaare überdacht werden. In CORDS wird bspw. als Reduktionsregel vorgeschlagen, nur Spaltenpaare zu be-

[4]Falls die betroffenen Spalten einen Zähler besitzen, der bei Änderungsoperationen hochgezählt wird (vgl. z. B. [1]), können natürlich auch solche Daten mit in die Wahl der Stichprobe einfließen, allerdings sind hier unterschiedliche „Ausgangszeiten“ zu beachten.

[5]Dadurch stören wir zwar etwas die vorgegebene Abarbeitungsreihenfolge der infrage kommenden Spaltenpaare, aber der Fall ist ja auch dringend.

trachten, die im Anfrageworkload vorkommen (dazu müssen von CORDS nur die Anfragen, aber nicht deren Ergebnisse betrachtet werden). Würde sich dann aber der Workload dahingehend ändern, dass völlig neue Spalten oder Tabellen abgefragt werden, hätten wir dasselbe Problem wie bei einem rein reaktiven Verfahren. Deswegen muss hier eine Zwischenlösung gefunden werden, die Spaltenkombinationen aus Anfragen bevorzugt „behandelt“, sich aber nicht darauf beschränkt.

Außerdem muss überlegt werden, wann wir Statistikdaten, die auf Stichproben beruhen, wieder löschen können. Im reaktiven Fall fiel die Entscheidung leicht aus, weil fehlender Zugriff auf die Daten auch ein fehlendes Nutzerinteresse widerspiegelt und auf diese Weise auch keine Aktualisierung mehr stattfindet, sodass die Metadaten irgendwann unbrauchbar werden.

Basieren die Statistiken dagegen auf Stichproben, müssen sie von Zeit zu Zeit aktualisiert werden. Passiert diese Aktualisierung ohne zusätzliche Überprüfung auf Korrelation (welche ja aufgrund geänderten Datenbestands nachlassen könnte), müssen mit der Zeit immer mehr zusätzliche Statistikdaten über Spaltenpaare gespeichert und gewartet werden. Der für Statistikdaten zur Verfügung stehende Speicherplatz im Katalog kann so an seine Grenzen treten, außerdem kostet die Wartung wiederum Kapazität des DBMS. Hier müssen sinnvolle Entscheidungen über die Wartung und das „Aufräumen“ nicht mehr benötigter Daten getroffen werden.

6. REFERENCES

[1] A. Aboulnaga, P. J. Haas, S. Lightstone, G. M. Lohman, V. Markl, I. Popivanov, and V. Raman. Automated statistics collection in DB2 UDB. In *VLDB*, pages 1146–1157, 2004.

[2] S. Babu, P. Bizarro, and D. J. DeWitt. Proactive re-optimization. In *SIGMOD Conference*, pages 107–118. ACM, 2005.

[3] E. Behm, V. Markl, P. Haas, and K. Murthy. Integrating query-feedback based statistics into informix dynamic server, Apr. 03 2008.

[4] P. Brown and P. J. Haas. BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. In *VLDB 2003: Proceedings of 29th International Conference on Very Large Data Bases, September 9–12, 2003, Berlin, Germany*, pages 668–679, 2003.

[5] N. Bruno, S. Chaudhuri, and L. Gravano. Stholes: a multidimensional workload-aware histogram. *SIGMOD Rec.*, 30(2):211–222, May 2001.

[6] P. J. Haas, F. Hueske, and V. Markl. Detecting attribute dependencies from query feedback. In *VLDB*, pages 830–841. ACM, 2007.

[7] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. CORDS: automatic discovery of correlations and soft functional dependencies. In ACM, editor, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data 2004, Paris, France, June 13–18, 2004*, pages 647–658, pub-ACM:adr, 2004. ACM Press.

[8] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik. Correlation maps: A compressed access method for exploiting soft functional dependencies. *PVLDB*, 2(1):1222–1233, 2009.

[9] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic. Robust query processing through progressive optimization. In ACM, editor, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data 2004, Paris, France, June 13–18, 2004*, pages 659–670. ACM Press, 2004.

[10] T. Neumann and C. Galindo-Legaria. Taking the edge off cardinality estimation errors using incremental execution. In *BTW*, pages 73–92, 2013.

[11] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. ISOMER: Consistent histogram construction using query feedback. In *ICDE*, page 39. IEEE Computer Society, 2006.

[12] M. Stillger, G. Lohman, V. Markl, and M. Kandil. LEO - DB2's learning optimizer. In *Proceedings of the 27th International Conference on Very Large Data Bases(VLDB '01)*, pages 19–28, Orlando, Sept. 2001.